# 🧾 Pre-Alert Performance Test – Full Summary (DC & DR)

## 🧩 1. Overview — What These Files Are

You uploaded **six CSV files** — they are results from **three different types of performance tests** for your **Pre-Alert system**, each performed on **two environments**:

| Test Type | Purpose | Files |
|---|---|---|
| **Load Test** | To verify that the system can handle **expected normal daily usage** steadily. | `loadtest_dc.csv`, `loadtest_dr.csv` |
| **Spike Test** | To see how the system reacts when **users suddenly surge** (like peak hours or promotions). | `spike_test_dc.csv`, `spike_test_dr.csv` |
| **Stress Test** | To push the system **beyond its normal capacity** and identify **breaking points** or recovery behavior. | `stress_test_dc.csv`, `stress_test_dr.csv` |

> 💡 Think of it like testing a car:
>
> - **Load test** → normal driving
> - **Spike test** → sudden acceleration
> - **Stress test** → redline to see limits

You ran these tests on two sites:

- **DC (Data Center)** – the primary production site
- **DR (Disaster Recovery)** – the backup/failover site

---

## ⚙️ 2. What the Tests Measure

All six files share similar columns that tell the story of performance:

| Column | Meaning (Simple) |
|---|---|
| **VUS (Virtual Users)** | Simulated users active during the test |
| **Requests** | Number of total actions sent by users |
| **Duration** | How long the test lasted |
| **Throughput (req/s)** | How many requests the system handled per second — higher is better |
| **Average Response Time (s)** | How long, on average, it took the system to respond — lower is better |
| **P90 / P95 Response (s)** | The slowest 10% and 5% of requests — helps identify lag spikes |
| **App CPU / Memory (%)** | How much server power and memory were used |
| **Pods / Connection / Timeout** | Infrastructure configuration details |
| **Remarks** | Notes from testers |

---

## 📊 3. Test Results Summary (Key Metrics)

| Test Type | Env | Avg Throughput (req/s) | Avg Response (s) | Peak Throughput | Avg CPU (%) |
|---|---|---|---|---|---|
| **Load** | DC | **27.5** | **4.23** | — | **9.1** |
| **Load** | DR | **16.6** | **5.25** | — | **18.8** |
| **Spike** | DC | **19.6** | **11.03** | **24.4** | **19.9** |
| **Spike** | DR | **17.0** | **11.68** | **20.5** | **19.3** |

| Test Type | Env | Avg Throughput (req/s) | Avg Response (s) | Peak Throughput | Avg CPU (%) |
|---|---|---|---|---|---|
| **Stress** | DC | **47.7** | **13.1** | **63.9** | **21.0** |
| **Stress** | DR | **45.9** | **12.0** | **50.5** | **24.4** |

## 🧠 4. Interpretation — What This Means

### ✅ DC (Primary Site)

- Handles **more load and higher throughput** in every test.
- Response times increase predictably under heavier load (normal behavior).
- CPU stays low (9–21%), meaning it's **not CPU-limited** — delays likely come from **database or network bottlenecks**.
- Under stress, it still runs stably, showing **good resilience**.

### ⚠️ DR (Disaster Recovery Site)

- Slightly **slower and lower throughput** than DC in all scenarios.
- Uses **more CPU** (18–24%) to achieve lower performance — may indicate smaller server size or less tuning.
- Stable but **not yet optimized** to match DC performance.

### 📈 Overall System Behavior

- Across all tests, **Pre-Alert stayed stable and responsive** — no major crashes or failures.
- However, **average response times (4–13 seconds)** are **too high** for smooth end-user experience.
- **CPU is low**, so the bottleneck is likely in:
  - Database queries or connection pool
  - I/O latency
  - Timeout or connection settings
  - Application-level thread waits
- **P90/P95** and **Memory data** are missing — we can't see "worst-case" delays or memory pressure.

## 🧩 5. What Each Test Tells You

| Test | What It Proves | Observed Result |
|---|---|---|
| **Load Test** | Can handle normal usage steadily | ✅ Stable at 500 users; DC faster |
| **Spike Test** | Can absorb sudden traffic surges | ⚠️ Stable but slow; response ~11s |
| **Stress Test** | System behavior under overload | ✅ Handles 1200 users; DC peaks at 63.9 req/s |

> 🧠 So: Pre-Alert can withstand load, absorb spikes, and stay up under stress — but needs performance tuning to improve speed.

## 🧱 6. Bottlenecks (Likely Causes of Slowness)

1. **Database connection or query delays** — slow SQL or too few DB connections.
2. **Timeout settings** — long waits can cause queue build-up.

3. **Insufficient scaling** — too few pods or auto-scaling thresholds too high.
4. **Network latency or I/O waits** — check API dependencies.
5. **Memory visibility missing** — can't confirm memory bottlenecks.

---

## 🧾 7. Recommendations

### 📊 Data Improvements

- Always include **P90 / P95 latency**, **Memory usage**, and **error counts** in next runs.
- Track **HTTP response codes** (2xx, 4xx, 5xx) to identify slow failures.

### ⚙️ System Tuning

- Optimize **database indexes** and increase **connection pool limits**.
- Tune **Nginx/app/DB timeouts** to reduce slow hangs.
- Check **pod limits** and consider **auto-scaling** at 80% CPU/memory.
- Aim for:
  - **Avg Response < 3 sec**
  - **P95 < 5 sec**
  - **CPU < 80%**
  - **Memory < 85%**

### 🧰 Infrastructure

- Ensure DR has **equal resource specs** as DC.
- Conduct **longer stability runs** (1–2 hours) to observe performance trends.

---

## 🏁 8. Overall Conclusion

✅ **Pre-Alertis stable under all loads** — it didn't crash or fail.

⚙️ **DC performs better** than DR in all metrics.

⚠️ **Response times are high**, so tuning and optimization are needed before production peak.

🚀 **Next Step:** Tune DB and timeouts, add auto-scaling, and re-run tests with percentile and memory metrics enabled.

---

## 💬 9. One-Page Summary

**Pre-Alert Performance Summary – DC & DR**

**Test Types:** Load • Spike • Stress

**Goal:** Measure stability and responsiveness under different load patterns.

| Test | Env | Avg Resp (s) | Avg Throughput (req/s) | CPU (%) | Summary |
|---|---|---|---|---|---|
| Load | DC | 4.2 | 27.5 | 9.1 | Stable, efficient |
| Load | DR | 5.3 | 16.6 | 18.8 | Slower, stable |
| Spike | DC | 11.0 | 19.6 | 19.9 | Handles surge OK |
| Spike | DR | 11.7 | 17.0 | 19.3 | Slightly slower recovery |

| Test | Env | Avg Resp (s) | Avg Throughput (req/s) | CPU (%) | Summary |
|---|---|---|---|---|---|
| Stress | DC | 13.1 | 47.7 | 21.0 | Strong capacity |
| Stress | DR | 12.0 | 45.9 | 24.4 | Stable, less headroom |

**Overall Findings:**

✅ Stable under all loads (no crash)

⚙️ DC > DR performance

⚠️ Slow responses (4–13s) → database or connection bottlenecks

❌ Missing P90/P95 + memory data limits analysis

**Next Steps:**

- Optimize DB queries and timeouts
- Enable auto-scaling
- Re-test with percentile metrics and memory tracking
- Target <3s average response for end-user satisfaction

**Final Verdict:**

> Pre-Alert is reliable but slow — it passes stability, but needs tuning and scaling to meet real-world performance standards.

---